Unit LCD

SKU:U120

Description

Unit LCD is a 1.14 inch color LCD expansion screen unit. It adopts ST7789V2 drive scheme, the resolution is 135*240, and it supports RGB666 display (262,144 colors). The internal integration of ESP32-PICO control core (built-in firmware, display development is more convenient), support through I2C (addr: 0x3E) communication interface for control and firmware upgrades. The back of the screen is integrated with a magnetic design, which can easily adsorb the metal surface for fixing. The LCD screen extension is suitable for embedding in various instruments or control devices that need to display simple content as a display panel.

Product Features

- 1.14 inch color LCD display panel
- ST7789V2 drive scheme
- I2C communication interface
- 135*240 resolution
- Viewing angle: full viewing angle
- Magnetic back design
- Support I2C firmware upgrade

Include

- 1x Unit LCD
- 1x HY2.0-4P Cable

Applications

- Information display

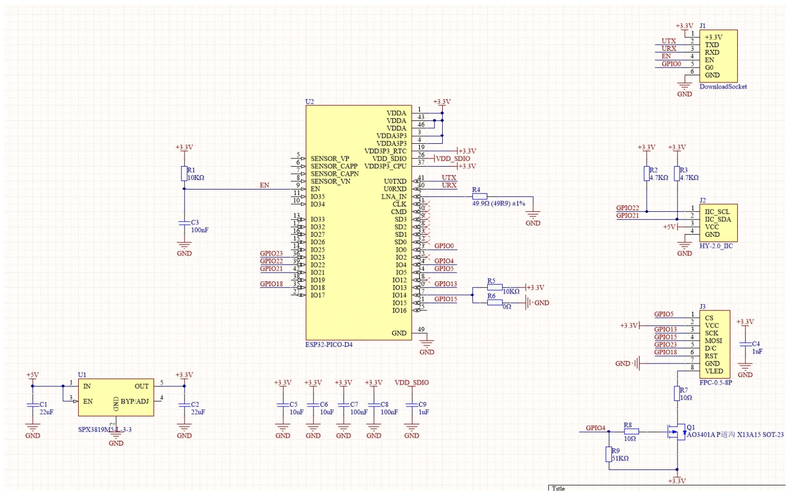
Specification

Specification	Parameter
Working current	45.7mA
Communication protocol	I2C address: 0x3E
Display size	1.14 inch
Pixel pitch	0.1101(H) x 0.1038(V) mm
Resolution	135*240
Viewable	Full view
Operating temperature	32°F to 104°F (0°C to 40°C)
Net weight	8.5g
Gross weight	20g
Product Size	48*24*8mm
Package Size	67*52*12.5mm
Case material	Plastic (PC)

PinMap

M5Core(PORT A)	GPIO22	GPIO21	5V	GND
Unit LCD	SCL	SDA	5V	GND

Schematic



Related Link

- **Datasheet**
  - ST7789V2
- **Library**
  - M5GFX Library

Example

- Arduino

```
#include <M5UnitLCD.h>

M5UnitLCD display;

M5Canvas canvas(&display);

static constexpr char text[] = "Hello world！こんにちは世界！ this is long long string sample. 寿限無、寿限無、五劫の擦り切れ、海砂利水魚の、水行末・雲来末・風来末、喰う寝る処に住む処、藪ら柑子の藪柑子、パイポ・パイポ・パイポのシューリンガン、シューリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナの、長久命の長助";
static constexpr size_t textlen = sizeof(text) / sizeof(text[0]);
int textpos = 0;
int scrollstep = 2;

void setup(void)
{
  display.init();
  display.setRotation(2);
  canvas.setColorDepth(1); // mono color
  canvas.setFont(&fonts::lgfxJapanMinchoP_32);
  canvas.setTextWrap(false);
  canvas.setTextSize(2);
  canvas.createSprite(display.width() + 64, 72);
}

void loop(void)
{
  int32_t cursor_x = canvas.getCursorX() - scrollstep;
  if (cursor_x <= 0)
  {
    textpos = 0;
    cursor_x = display.width();
  }

  canvas.setCursor(cursor_x, 0);
  canvas.scroll(-scrollstep, 0);
  while (textpos < textlen && cursor_x <= display.width())
  {
    canvas.print(text[textpos++]);
    cursor_x = canvas.getCursorX();
  }
  display.waitDisplay();
  canvas.pushSprite(&display, 0, (display.height() - canvas.height()) >> 1);
}
```

About Unit LCD

- Unit LCD is an I2C unit with ESP32 and ST7789V2.
- It has an IPS panel with a resolution of 135 x 240.
- The number of colors that can be displayed is 262,144 colors of RGB666, which is the specification of ST7789V2.

- The ESP32 is in charge of I2C communication and draws in the frame buffer in memory based on the received contents.
- The contents of the frame buffer in the memory of the ESP32 are reflected in ST7789V2 by DMA transfer of SPI communication.

- It is represented by RGB888 16,777,216 colors on the framebuffer.

About Communication with Unit LCD

- You can send commands and receive data to the Unit LCD using I2C communication.
- The maximum communication speed of I2C communication is 400kHz.
- The initial value of the 7-bit address of I2C is 0x3E. It can be changed with the CHANGE_ADDR command.
- The required number of bytes to send depends on the command.
  Some commands complete in 1 Byte, while others require 7 Byte.
  There are also indefinite length commands that do not end until communication is stopped.
- If I2C communication STOP or RESTART occurs during command transmission, the interrupted command will not be processed.
  It must be transmitted uninterrupted to the end in a single transmission sequence.
- After sending a fixed-length command, you can send another command in succession.
- After sending an indefinite-length command, I2C communication must be stopped to indicate the end of the command.
- If you send a NOP command or an undefined command, the communication content is ignored until I2C communication stops.
- Since the I2C communication unit and the drawing processing unit operate in parallel, I2C communication can be performed even during the drawing processing.
- The I2C communication contents are stored in the command buffer on the ESP32 memory, and the drawing processing unit processes this sequentially.
- You should use the READ_BUFCOUNT command to check the remaining amount of the buffer , as sending a large amount of heavy processing, such as extensive fill or range copy, can overwhelm the command buffer.

About drawing commands

- You can draw a rectangle by filling any area with a single color with FILLRECT.
- If you want to draw only one pixel, you can use DRAWPIXEL instead of FILLRECT.
- If you use a command that omits the foreground color, the last used color is used.
- You can specify the drawing range with CASET and RASET and send the image data with the WRITE_RAW command.
- You can use WRITE_RLE instead of WRITE_RAW to send run-length compressed image data.

Command list

※ Undefined commands are treated as NOP.

hex	len	command	description	send params
0x00	1-∞	NOP	Do nothing until communication stops	[0] 0x00 [1-∞] Ignored value
0x20	1	INVOFF	Disable color inversion	[0] 0x20
0x21	1	INVON	Enable color inversion	[0] 0x21
0x22	2	BRIGHTNESS	Backlight brightness setting 0:Off - 255:Full lights	[0] 0x22 [1] Brightness(0-255)
				[0] 0x23

hex	len	command	description	send params
0x23	7	COPYRECT	Rectangle range copy	[0] 0x23 [1] Copy source X_Left [2] Copy source Y_Top [3] Copy source X_Right [4] Copy source Y_Bottom [5] Copy destination X_Left [6] Copy destination Y_Top
0x2A	3	CASET	X-direction range selection	[0] 0x2A [1] X_Left [2] X_Right
0x2B	3	RASET	Y-direction range selection	[0] 0x2B [1] Y_Top [2] Y_Bottom
0x36	2	ROTATE	Set drawing orientation 0:Normal / 1:90° / 2:180° / 3:270° 4-7:flips 0-3 upside down	[0] 0x36 [1] Setting value (0-7)
0x38	2	SET_POWER	Operating speed setting (power consumption setting) 0:Low speed / 1:Normal / 2:High speed	[0] 0x38 [1] Setting value (0-2)
0x39	2	SET_SLEEP	LCD panel sleep setting 0:wake up / 1:sleep	[0] 0x39 [1] Setting value (0-1)
0x41	2-∞	WRITE_RAW_8	draw image RGB332	[0] 0x41 [1] RGB332 until [1] communication STOP.
0x42	3-∞	WRITE_RAW_16	draw image RGB565	[0] 0x42 [1-2] RGB565 until [1-2] communication STOP.
0x43	4-∞	WRITE_RAW_24	draw image RGB888	[0] 0x43 [1-3] RGB888 until [1-3] communication STOP.
0x44	5-∞	WRITE_RAW_32	draw image ARGB8888	[0] 0x44 [1-4] ARGB8888 until [1-4] communication STOP.

hex	len	command	description	send params
				until [1-4] communication STOP.
			draw image A8	[0] 0x45
0x45	2-∞	WRITE_RAW_A	only alpha channel. Use the last used drawing color.	[1] A8 until [1] communication STOP.
0x49	3-∞	WRITE_RLE_8	draw RLE image RGB332	[0] 0x49 [1-∞] RLE Data
0x4A	4-∞	WRITE_RLE_16	draw RLE image RGB565	[0] 0x4A [1-∞] RLE Data
0x4B	5-∞	WRITE_RLE_24	draw RLE image RGB888	[0] 0x4B [1-∞] RLE Data
0x4C	6-∞	WRITE_RLE_32	draw RLE image ARGB8888	[0] 0x4C [1-∞] RLE Data
0x4D	3-∞	WRITE_RLE_A	draw RLE image A8 only alpha channel. Use the last used drawing color.	[0] 0x4D [1-∞] RLE Data
0x50	1	RAM_FILL	Fill the selection with the last used drawing color	[0] 0x50
0x51	2	SET_COLOR_8	Specify the drawing color with RGB332	[0] 0x51 [1] RGB332
0x52	3	SET_COLOR_16	Specify the drawing color with RGB565	[0] 0x52 [1-2] RGB565
0x53	4	SET_COLOR_24	Specify the drawing color with RGB888	[0] 0x53 [1-3] RGB888
0x54	5	SET_COLOR_32	Specify the drawing color with ARGB8888	[0] 0x54 [1-4] ARGB8888
0x60	3	DRAWPIXEL	Draw single dot Use the drawing color that is stored	[0] 0x60 [1] X [2] Y
0x61	4	DRAWPIXEL_8	Draw single dot RGB332 1Byte for drawing color specification	[0] 0x61 [1] X [2] Y [3] RGB332
0x62	5	DRAWPIXEL_16	Draw single dot RGB565 2Byte for drawing color specification	[0] 0x62 [1] X [2] Y

hex	len	command	description	send params
			RGB565 2Byte for drawing color specification	[2] Y [3-4] RGB565
0x63	6	DRAWPIXEL_24	Draw single dot RGB888 3Byte for drawing color specification	[0] 0x63 [1] X [2] Y [3-5] RGB888
0x64	7	DRAWPIXEL_32	Draw single dot ARGB8888 4Byte for drawing color specification Transparent composition with existing drawing contents	[0] 0x64 [1] X [2] Y [3-6] ARGB8888
0x68	5	FILLRECT	Fill rectangle Use the drawing color that is stored	[0] 0x68 [1] X_Left [2] Y_Top [3] X_Right [4] Y_Bottom
0x69	6	FILLRECT_8	Fill rectangle RGB332 1Byte for drawing color specification	[0] 0x69 [1] X_Left [2] Y_Top [3] X_Right [4] Y_Bottom [5] RGB332
0x6A	7	FILLRECT_16	Fill rectangle RGB565 2Byte for drawing color specification	[0] 0x6A [1] X_Left [2] Y_Top [3] X_Right [4] Y_Bottom [5-6] RGB565
			Fill rectangle	[0] 0x6B [1] X_Left [2] Y_Top

0x6B	18	FILLRECT_24	Fill rectangle **description** RGB888 3Byte for drawing color specification	[2] Y_Top **send params** [3] X_Right [4] Y_Bottom [5-7] RGB888
0x6C	9	FILLRECT_32	Fill rectangle ARGB8888 4Byte for drawing color specification Transparent composition with existing drawing contents	[0] 0x6C [1] X_Left [2] Y_Top [3] X_Right [4] Y_Bottom [5-8] ARGB8888
0xA0	4	CHANGE_ADDR	I2C address change. prevent unintended execution, [2] specifies the bit inversion value of [1].	[0] 0xA0 [1] new I2C address. [2] Bit inversion of [1] [3] 0xA0

Command list (readable commands)

hex	len	command	description	return values
0x04	1	READ_ID	ID and firmware version. 4Byte received	[0] 0x77 [1] 0x89 [2] Major version [3] Minor version
0x09	1	READ_BUFCOUNT	Get remaining command buffer. The higher the value, the more room there is. Can be read out continuously.	[0] remaining command buffer (0~255) Repeated reception is possible.
0x81	1	READ_RAW_8	Readout of RGB332 image	[0] RGB332 Repeat [0] until communication STOP.
0x82	1	READ_RAW_16	Readout of RGB565 image	[0-1] RGB565 Repeat [0-1] until communication STOP.
0x83	1	READ_RAW_24	Readout of RGB888 image	[0-2] RGB888 Repeat [0-2] until communication STOP.

Communication example

Example: Use the Fill Rectangle command 0x6A to fill the rectangle range of X16-31 and Y32-47 with red.

index	hex	description
0	0x6A	Fill rectangle RGB565
1	0x10	X Left
2	0x20	Y Top
3	0x1F	X Right
4	0x2F	Y Bottom
5	0xF8	Color data RRRRRGGG(red)
6	0x00	Color data GGGBBBBB(red)

The command 6Ah is a total of 7Byte command sequence.
If an I2C communication STOP or RESTART occurs during transmission, the command will not be processed. It is necessary to transmit without interruption until the end in a single transmission sequence.
Any of the rectangle fill commands 68h to 6Ch can be used for rectangle fill. Indexes 1 through 4 are the same, but indexes 5 and onward have different methods for specifying colors.
The "remembered color" of command 68h means that the last specified color will be reused. In other words, if you want to do several rectangular fills of the same color in succession, you can specify the color only for the first rectangular fill and then omit the color specification by using the 68h command.
Command 6Ch, ARGB8888, allows you to specify an alpha channel (transparency), which allows you to combine the already drawn content with the drawing color.

Example: Using the range specification command 0x2A/0x2B and the image transmission command, draw an image in the rectangular range of X 10 to 13 and Y 14 to 17.

index	hex	description
0	0x2A	X-direction range selection
1	0x0A	X Left(10)
2	0x0D	X Right(13)
3	0x2B	Y-direction range selection
4	0x0E	Y Top(14)
5	0x11	Y Bottom(17)
6	0x43	Draw image RGB888
7-54	??	Image data(RGB888 ×16)

Example: Sending an RLE (run length encoding) image using the WRITE_RLE command.

- The RLE specification is based on the RLE for BMP files.
- Unlike RLE for BMP files, it can be used for RGB565 and RGB888.
- It first sends a consecutive number of pixels of the same color (0-255), followed by the color data.
- If 0 is sent to a consecutive number, it will be in direct mode without using RLE.

- In direct mode, the number of pixels (1-255) is sent first, followed by the color data for the number of pixels.

index	hex	description
0	0x4A	Draw RLE image RGB565
1	0x07	Consecutive number (7pixel)
2-3	0xF800	Color data (red)
4	0x00	Consecutive number (0pixel) switch to direct mode
5	0x03	Consecutive number of direct mode(3pixel)
6-7	0x07E0	Color data (green)
8-9	0x001F	Color data (blue)
10-11	0xF800	Color data (red)
12	0x04	Consecutive number (4pixel)
13-14	0x001F	Color data (blue)

The above example will be handled as follows.

- index1-3 : Draw 7 pixels of red in RLE mode.
- index4-5 : Switch to direct mode and instruct it to draw 3 pixels.
- index6-11: Sends the color for 3 pixels in direct mode and draws green, blue, and red.
- Since the direct mode for 3 pixels is completed by index11, we will return to the RLE mode from index12.
- index12-14 : Draws 4 pixels of blue in RLE mode.

Video